

Mapping distributional to model-theoretic semantic spaces: a baseline

Franck Deroncourt

MIT

francky@mit.edu

Abstract

Word embeddings have been shown to be useful across state-of-the-art systems in many natural language processing tasks, ranging from question answering systems to dependency parsing. (Herbelot and Vecchi, 2015) explored word embeddings and their utility for modeling language semantics. In particular, they presented an approach to automatically map a standard distributional semantic space onto a set-theoretic model using partial least squares regression. We show in this paper that a simple baseline achieves a +51% relative improvement compared to their model on one of the two datasets they used, and yields competitive results on the second dataset.

1 Introduction

Word embeddings are one of the main components in many state-of-the-art systems for natural language processing (NLP), such as language modeling (Mikolov et al., 2010), text classification (Socher et al., 2013; Kim, 2014; Blunsom et al., 2014; Lee and Deroncourt, 2016), question answering (Weston et al., 2015; Wang and Nyberg, 2015), machine translation (Bahdanau et al., 2014; Tamura et al., 2014; Sundermeyer et al., 2014), as well as named entity recognition (Collobert et al., 2011; Deroncourt et al., 2016; Lample et al., 2016; Labeau et al., 2015).

Word embeddings can be pre-trained using large unlabeled datasets typically based on token co-occurrences (Mikolov et al., 2013; Collobert et al., 2011; Pennington et al., 2014). They can also be jointly learned with the task.

Understanding what information word embeddings contain is subsequently of high interest. (Herbelot and Vecchi, 2015) investigated a method to map word embeddings to formal semantics, which is the center of interest of this paper. Specifically, given a feature and a word vector of a concept, they tried to automatically find how often the given concept has the given feature. For example, the concept *yam* is always a *vegetable*, the concept *cat* has a coat most of the time, the concept *plug* has sometimes 3 prongs, and the concept *dog* never has wings.

The method they used was based on partial least squares regression (PLSR). We propose a simple baseline that outperforms their model.

2 Task

In this section, we summarize the task presented in (Herbelot and Vecchi, 2015). The following is an example of a concept along with some of its features, as formatted in one of the two datasets used to evaluate the model:

yam	a_vegetable	all	all	all
yam	eaten_by_cooking	all	most	most
yam	grows_in_the_ground	all	all	all
yam	is_edible	all	most	all
yam	is_orange	some	most	most
yam	like_a_potato	all	all	all

The concept *yam* has six features (*a_vegetable*, *eaten_by_cooking*, *grows_in_the_ground*, *is_edible*, *is_orange*, and *like_a_potato*). Each feature in this dataset is annotated by three different humans. The annotation is a quantifier that reflects how frequently the concept has a feature. Five quantifiers are used:

no, *few*, *some*, *most*, and *all*. In this example, the concept *yam* has been annotated as *some*, *most* and *most* for the feature *is_orange*.

Each of the five quantifiers is converted into a numerical format with the following (somehow arbitrary) mapping: *no* \mapsto 0; *few* \mapsto 0.05; *some* \mapsto 0.35; *most* \mapsto 0.95; *all* \mapsto 1. The value is averaged over the three annotators. Using this mapping, we can map a concept into a “model-theoretic vector” (also called feature vector). If a feature has not been annotated for a concept, then the element in the model-theoretic vector corresponding to the feature will have value 0. As a result, any element of a model-theoretic vector that has value 0 may correspond to a feature that has either been annotated as *no* by the three annotators, or not been annotated (presumed *no*). Given that there can be many features and it is possible that only some of them are annotated for each concept, the model-theoretic vector may be quite sparse.

In the *yam* example, if we only included features annotated with *yam*, the model-theoretic vector would be as follows:

$$\begin{bmatrix} \frac{\text{all}+\text{all}+\text{all}}{3} \\ \frac{\text{all}+\text{most}+\text{most}}{3} \\ \frac{\text{all}+\text{all}+\text{all}}{3} \\ \frac{\text{all}+\text{most}+\text{all}}{3} \\ \frac{\text{some}+\text{most}+\text{most}}{3} \\ \frac{1+1+1}{3} \end{bmatrix} = \begin{bmatrix} \frac{1+1+1}{3} \\ \frac{1+0.95+0.95}{3} \\ \frac{1+1+1}{3} \\ \frac{1+0.95+1}{3} \\ \frac{0.35+0.95+0.95}{3} \\ \frac{1+1+1}{3} \end{bmatrix} \approx \begin{bmatrix} 1 \\ 0.967 \\ 1 \\ 0.983 \\ 0.75 \\ 1 \end{bmatrix}$$

The additional coordinates corresponding to all the remaining features would be zero. Each concept word will have a vector of the same dimension (number of unique features) in the same dataset. The coordinates mean the same from one concept to another. For example, the feature *is_vegetable* appears in the same coordinate position in all the vectors.

3 Datasets

Two datasets are used:

- The Animal Dataset (AD) (Herbelot, 2013) contains 73 concepts and 54 features. All concepts are animals, and for each concept all

features are annotated by 1 human annotator. There are 3942 annotated pairs of concept-feature ($73 * 54 = 3942$). The dimension of the model-theoretic vectors will therefore be 54.

- TheMcRae norms (QMR) (McRae et al., 2005) contains 541 concepts covering living and non-living entities (e.g., alligator, chair, accordion), as well as 2201 features. One concept is annotated with 11.4 features on average by 3 human annotators. There are 6187 annotated pairs of concept-feature ($541 * 11.4 \approx 6187$). The dimension of the model-theoretic vectors will therefore be 2201, and each model-theoretic vector will have on average $2201 - 11.4 = 2189.6$ elements set to 0 due to unannotated features.

4 Model

In the previous section, we have seen how to convert a concept into a model-theoretic vector based on human annotations. The goal of (Herbelot and Vecchi, 2015) is to analyze whether there exists a transformation from the word embedding of a concept to its model-theoretic vector, the gold standard being the human annotations. The word embeddings are taken from the word embeddings pre-trained with word2vec *GoogleNews-vectors-negative300*¹ (300 dimensions), which were trained on part of the Google News dataset, consisting of approximately 100 billion words.

The transformation used in (Herbelot and Vecchi, 2015) is based on Partial Least Squares Regression (PLSR). The PLSR is fitted on the training set: the inputs are the word embeddings for each concept, and the outputs are the model-theoretic vectors for each concept.

To assess the quality of the predictions, the Spearman rank-order correlation coefficient is computed between the predictions and the gold model-theoretic vectors, ignoring all features for which a concept has not been annotated. The idea is that some of the features might be present but not given as options during annotation. The method should therefore not be penalized for not suggesting them. Figure 1 illustrates the model.

¹<https://code.google.com/p/word2vec/>

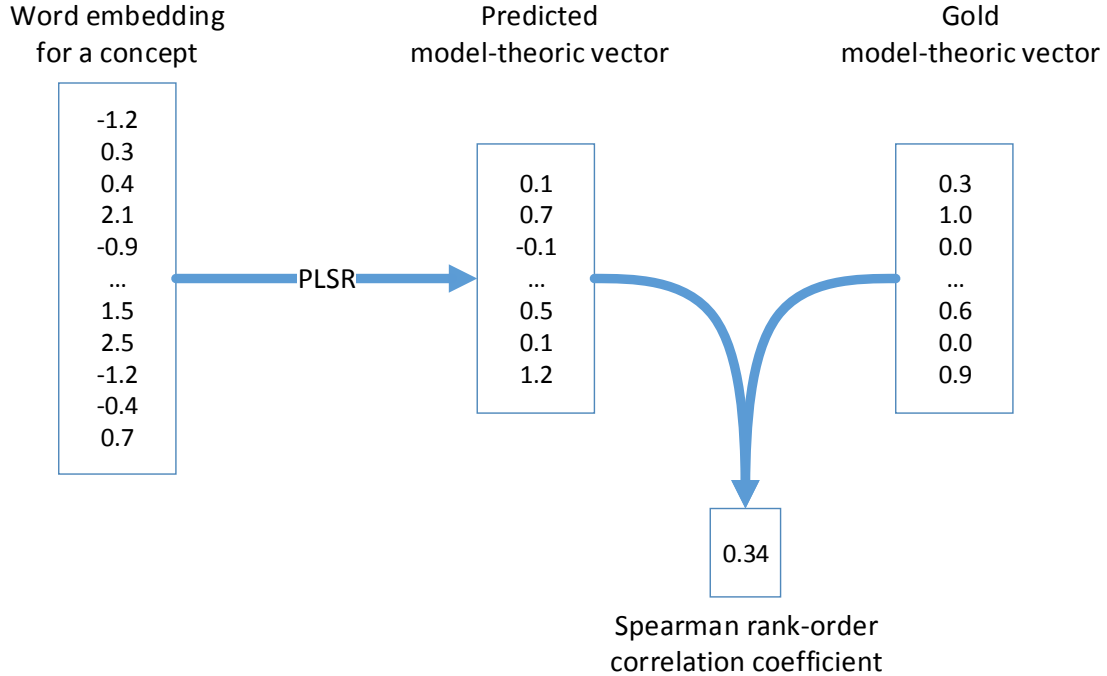


Figure 1: Overview of (Herbelot and Vecchi, 2015)’s system. The word embedding of a concept is transformed to a model-theoretic vector via a PLSR. The quality of the predicted model-theoretic vector is assessed with the Spearman rank-order correlation coefficient between the predictions and the gold model-theoretic vectors. Note that some of the elements that equal 0 in the gold model-theoretic vector may correspond to features that are not annotated for the concept. Such features are omitted when evaluating the Spearman rank-order correlation coefficient. Also, the dimension of the model-theoretic vectors could be larger or smaller than the dimension of the word embedding. Since the word embeddings we use have 300 dimensions, the model-theoretic vectors will be smaller than the word embeddings in the AD dataset, and larger in the QMR dataset.

5 Experiments

We compare (Herbelot and Vecchi, 2015)’s model (PLSR + word2vec) against three baselines: random vectors, mode, and nearest neighbor.

- *Mode*: A predictor that outputs, for each feature, the most common feature value (i.e., the mode) in the training set. For example, if a feature is annotated as *all* for most concepts, then the predictor will always output *all* for this feature. When finding the most common value of a feature, we ignore all the concepts for which the feature is not annotated. The resulting predictor does not take any concept into account when making a prediction. Indeed, the predicted values are always the same, regardless of the concept. If a feature has the same value for most concepts, the predictor may perform reasonably well.
- *Nearest neighbor (NN)*: A predictor that outputs for any concept the model-theoretic vector

from the training set corresponding to the most similar concept in the training set. Similarity is based on the cosine similarity of the word vectors. This is a simple nearest neighbor predictor.

- *Random vectors*: (Herbelot and Vecchi, 2015) used pre-trained word embeddings as input to the PLSR, we instead simply use random vectors of same dimension (300, continuous uniform distribution between 0 and 1).

We also apply retrofitting (Faruqui et al., 2014) on the word embeddings in order to leverage relational information from semantic lexicons by encouraging linked words to have similar vector representations. Using (Faruqui et al., 2014)’s retrofitting tool², we retrofit the word embeddings (*GoogleNews-vectors-negative300*) on each of the 4 datasets present in the retrofitting tool (*framenet*, *ppdb-xl*, *wordnet-synonyms+*, and *wordnet-synonyms*).

²<https://github.com/mfaruqui/retrofitting>

	AD			QMR		
	Min	Average	Max	Min	Average	Max
PLSR + word2vec	0.435	0.572	0.713	0.244	0.332	0.407
PLSR + word2vec + framenet	0.423	0.577	0.710	0.236	0.331	0.410
PLSR + word2vec + ppdb	0.455	0.583	0.688	0.247	0.332	0.421
PLSR + word2vec + wordnet	0.429	0.583	0.713	0.252	0.339	0.444
PLSR + word2vec + wordnet+	0.453	0.604	0.724	0.261	0.344	0.428
PLSR + random vectors	0.253	0.419	0.550	−0.017	0.087	0.178
NN + word2vec	0.338	0.524	0.751	0.109	0.215	0.291
NN + word2vec + framenet	0.321	0.516	0.673	0.108	0.204	0.288
NN + word2vec + ppdb	0.360	0.531	0.730	0.114	0.213	0.300
NN + word2vec + wordnet	0.384	0.551	0.708	0.115	0.208	0.297
NN + word2vec + wordnet+	0.390	0.597	0.806	0.138	0.235	0.324
NN + random vectors	0.244	0.400	0.597	−0.063	0.029	0.107
mode	0.432	0.554	0.643	0.420	0.522	0.605
true-mode	0.419	0.551	0.637	0.379	0.466	0.551
(Herbelot and Vecchi, 2015) (PLSR + word2vec)	?	0.634	?	?	0.346	?

Table 1: All the presented results are averaged over 1000 runs, except for the results of (Herbelot and Vecchi, 2015) in the last row. PLSR stands for partial least squares regression, NN for nearest neighbor, ppdb for the Paraphrase Database (Ganitkevitch et al., 2013). There are two ways to compute the mode: either taking the mode of the means of the 3 annotations (*mode*), or the mode for all annotations (*true-mode*). QMR has 3 potentially different annotations for each concept-feature pair, while AD has 3 only one annotation for each concept-feature pair: as a result, *mode* and *true-mode* have similar results for AD, but potentially different results for QMR. For each run, a train/test split was randomly chosen (60 training samples for AD, 400 for QMR, in order to have the same number of training samples as in (Herbelot and Vecchi, 2015)’s Table 2).

6 Results and discussion

Table 1 presents the results, using the Spearman correlation as the performance metric. The experiment was coded in Python using scikit-learn (Pedregosa et al., 2011) and the source as well as the complete result log and the two datasets are available online³. We could reproduce the results for the QMR dataset using PLSR and word2vec embeddings (0.346 in (Herbelot and Vecchi, 2015) vs. 0.332 in our experiments, but we could not ex-

actly reproduce the results for the AD dataset (0.634 in (Herbelot and Vecchi, 2015) vs. 0.572 in our experiments): this discrepancy most likely results from the choice of the training set. Our experiments’ results are averaged over 1000 runs, and for each run the training/test split is randomly chosen, the only constraint being having the same number of training samples as in (Herbelot and Vecchi, 2015). For the AD dataset, our worst run achieved 0.435, and our best run achieved 0.713, which emphasizes the lack of robustness of the results with respect to the train/test split. The variability is much lower for the

³<https://github.com/Franck-Dernoncourt/model-theoretic>

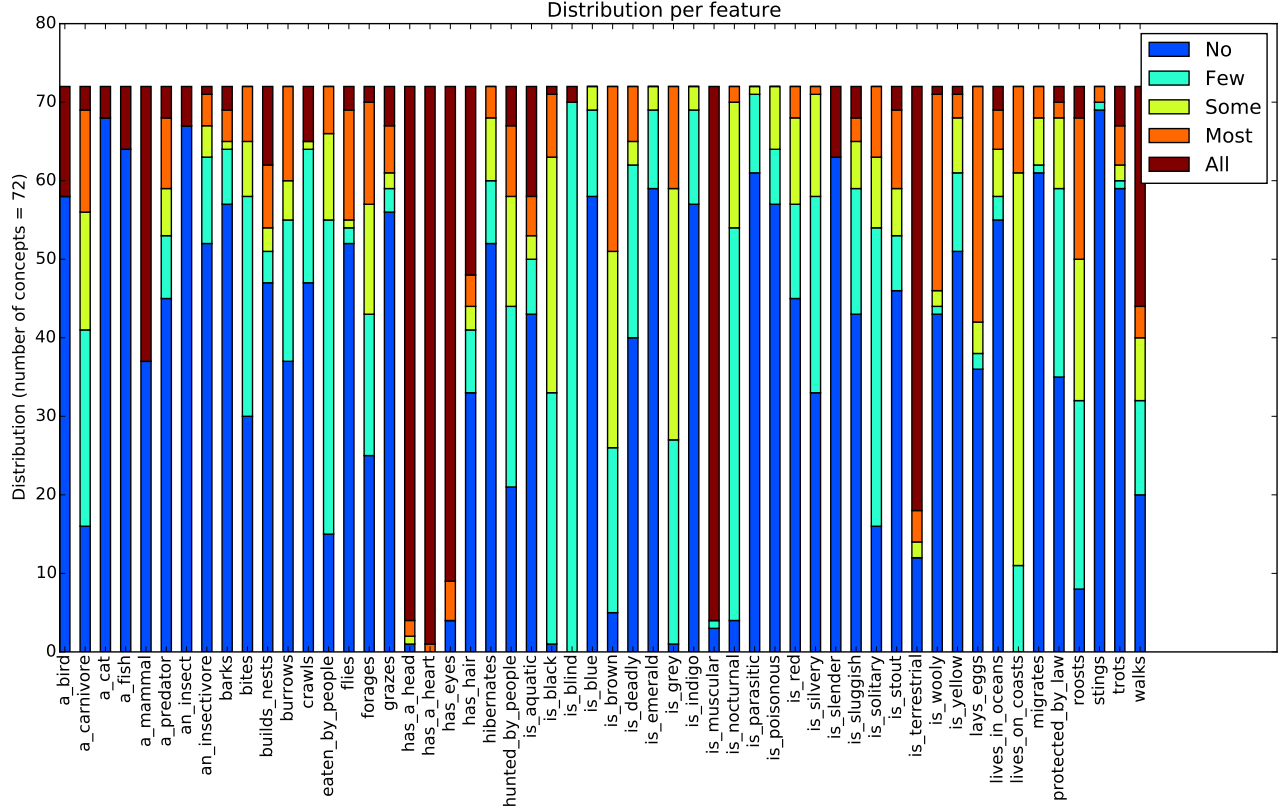


Figure 2: Stacked bars showing the distribution of quantifiers among features in the AD dataset: most features tend to have one clearly dominant quantifier. For example, the feature *a_cat* is almost always annotated with the quantifier *no*.

QMR dataset (min: 0.244; max: 0.407), which is expected since QMR is significantly larger than AD.

Furthermore, the *mode* baseline yields results that are good on the AD dataset (0.554, vs. 0.634 in (Herbelot and Vecchi, 2015) vs. 0.572 in our PLSR + word2vec implementation), and significantly better than all other models on the QMR dataset (0.522, vs. 0.346 in (Herbelot and Vecchi, 2015), i.e. +51% improvement). To get an intuition of why the mode baseline works well, Figures 2 and 3 show that most features tend to have one clearly dominant quantifier in the AD dataset. A similar trend can be found in the QMR dataset. In the AD dataset, there are 54 features, each of them being annotated for all 73 concepts. In the QMR dataset, there are 2201 features, each of them being annotated for only $\frac{6187}{2201} \approx 2.81$ concepts on average. As a result, it is much more difficult for the PLSR to learn the mapping from word embeddings to model-theoretic vectors in the QMR dataset than in the AD dataset. This explains why the *mode* baseline outperforms PLSR in the

QMR dataset but not in the AD dataset.

The *random vector* baseline with PLSR performs mediocrely on the AD dataset, and very poorly on the QMR dataset. The *nearest neighbor* baseline yields some competitive results on the AD dataset, but lower results on the QMR dataset. Lastly, using retrofitting increases the performances on both AD and QMR datasets. This is expected as applying retrofitting to word embeddings leverages relational information from semantic lexicons by encouraging linked words to have similar vector representations.

7 Conclusion

In this paper we have presented several baselines for mapping distributional to model-theoretic semantic spaces. The *mode* baseline significantly outperforms (Herbelot and Vecchi, 2015)’s model on the QMR dataset, and yields competitive results on the AD dataset. This indicates that state-of-the-art models do not efficiently map word embeddings to model-theoretic vectors in these datasets.

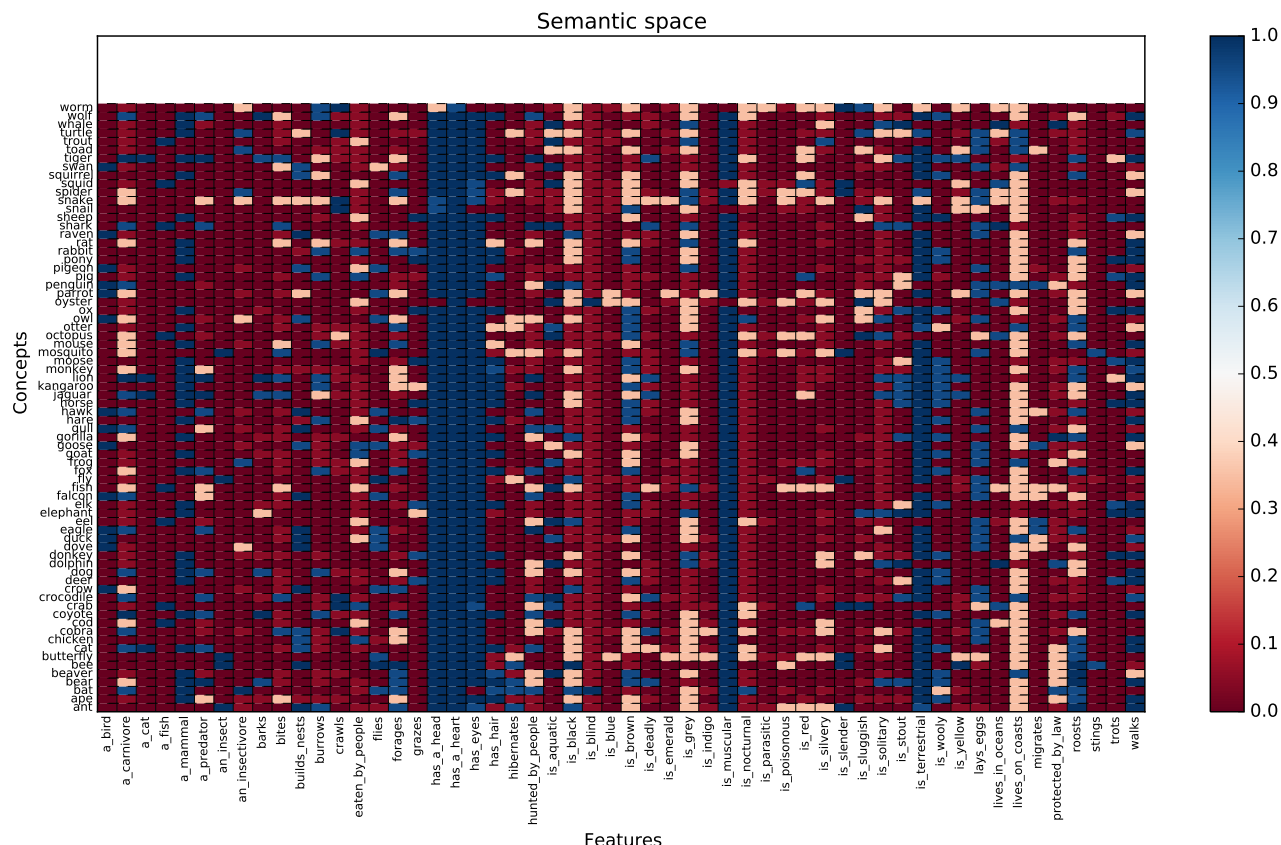


Figure 3: Heatmap showing the distribution of quantifiers among features in the AD dataset: most features tend to have one clearly dominant quantifier. The values of the heatmap are given by the following quantifier-scalar mapping: no \mapsto 0; few \mapsto 0.05; some \mapsto 0.35; most \mapsto 0.95; all \mapsto 1.

References

- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.
- Phil Blunsom, Edward Grefenstette, Nal Kalchbrenner, et al. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics.
- Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.
- Franck Dernoncourt, Ji Young Lee, Ozlem Uzuner, and Peter Szolovits. 2016. De-identification of patient notes with recurrent neural networks. *arXiv preprint arXiv:1606.03475*.
- Manaal Faruqui, Jesse Dodge, Sujay K Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2014. Retrofitting word vectors to semantic lexicons. *arXiv preprint arXiv:1411.4166*.
- Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proceedings of NAACL-HLT*, pages 758–764, Atlanta, Georgia, June. Association for Computational Linguistics.
- Aurélien Herbelot and Eva Maria Vecchi. 2015. Building a shared world: Mapping distributional to model-theoretic semantic spaces. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 22–32.
- Aurélien Herbelot. 2013. What is in a text, what isn't, and what this has to do with lexical semantics. *Proceedings of the Tenth International Conference on Computational Semantics (IWCS2013)*.
- Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1746–1751. Association for Computational Linguistics.

- Matthieu Labeau, Kevin Löser, and Alexandre Allauzen. 2015. Non-lexical neural architecture for fine-grained POS tagging. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 232–237, Lisbon, Portugal, September. Association for Computational Linguistics.
- Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.
- Ji Young Lee and Franck Dernoncourt. 2016. Sequential short-text classification with recurrent and convolutional neural networks. In *Human Language Technologies 2016: The Conference of the North American Chapter of the Association for Computational Linguistics, NAACL HLT 2016*.
- Ken McRae, George S Cree, Mark S Seidenberg, and Chris McNorgan. 2005. Semantic feature production norms for a large set of living and nonliving things. *Behavior research methods*, 37(4):547–559.
- Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTER-SPEECH*, volume 2, page 3.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.
- Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830.
- Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. GloVe: global vectors for word representation. *Proceedings of the Empirical Methods in Natural Language Processing (EMNLP 2014)*, 12:1532–1543.
- Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer.
- Martin Sundermeyer, Tamer Alkhouli, Joern Wuebker, and Hermann Ney. 2014. Translation modeling with bidirectional recurrent neural networks. In *EMNLP*, pages 14–25.
- Akihiro Tamura, Taro Watanabe, and Eiichiro Sumita. 2014. Recurrent neural networks for word alignment model. In *ACL (1)*, pages 1470–1480.
- Di Wang and Eric Nyberg. 2015. A long short-term memory model for answer sentence selection in question answering. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 707–712, Beijing, China, July. Association for Computational Linguistics.
- Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. 2015. Towards AI-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*.